The Qualification of Software Development Tools From the DO-178B Certification Perspective

Dr. Andrew J. Kornecki
Embry Riddle Aeronautical University

Dr. Janusz Zalewski
Florida Gulf Coast University

Software development tools are in wide use among safety-critical system developers. Examples of such use include aviation, automotive, space, nuclear, railroad, medical, and military applications. However, verification of tool output to ensure safety, mandated in highly regulated industries, requires enormous effort. If a tool is qualified, this effort can be reduced or even eliminated. The Radio Technical Commission for Aeronautics Document Order-178B and related documents provide guidelines by which to qualify these tools. However, current regulations, business models, and industry practice make this goal difficult to accomplish. This article discusses the qualification of development tools and the potential impact of this process on the aviation industry.

Software development tools are computer programs that help developers create other programs. Such tools have been in use since the early days of computing to improve the efficiency of the development process by automating mundane translation operations and bringing the level of abstraction closer to the application engineer. Nowadays, development tools are used in a variety of safety-critical applications, including the aviation, automotive, space, nuclear, railroad, medical, and military industries, and contribute to the risks associated with using respective products. Despite these risks to society, development tools are rarely qualified in a sense comparable to product certification in regulated industries. The objective of this article is to look at the current state of the tool qualification process, identify the issues, and propose recommendations for potential improvement, focusing on the aviation industry.

System Certification Versus Software Tool Qualification

Certification of airborne equipment is typically achieved through the Federal Aviation Administration (FAA) authorization of a type certificate (the entire aircraft), supplemental type certificate (new equipment in a specific aircraft), or a technical standard order (minimum performance standard for materials, parts, and appliances used on civil aircraft). A special committee (SC-145) of the Radio Technical Commission for Aeronautics (RTCA) convened in 1980 to establish guidelines for developing airborne systems. The report "Software Considerations in Airborne Systems and Equipment Certification" was published in January 1982 as the RTCA Document Order (DO)-178 (and revised as DO-178A in 1985).

Due to rapid advances in technology, the RTCA established a new committee (SC-167) in 1989 with the objective of updating the DO-178A by focusing on five areas: documentation integration and production, system issues, software development, software verification, and software configuration management and software quality assurance. The resulting document, DO-178B, provides guidelines for applicants developing software-intensive airborne systems [1, 2]. It discusses objectives that need to be met to show that the software development process provides specified levels of safety assurance. It also describes the processes and means of compliance.

Systems are categorized by DO-178B as meeting safety assurance levels A through E based on their criticality in supporting safe aircraft flight. The level A system is the most critical: The failure of such a system could result in a catastrophic failure condition for the aircraft. The level E system is the least critical: Such a system has no effect on the operational capability of the aircraft or pilot workload. Although the RTCA DO-178B is the leading source of guidelines for software developers engaged in such system construction, two other documents have critical bearing on the subject. RTCA DO-248B [3] clarifies some of the misinterpretation of the DO-178B. The FAA Order 8110.49 compiles a variety of guidelines related to the use of software in airborne systems. Chapter 9 is specifically dedicated to tool qualification [4].

A key component of the updated version of DO-178B is the concept of tool qualification elaborated in Section 12. Qualification is a supplementary process that the applicant may elect to follow in the course of certifying an airborne system. According to the definition given in DO-178B, tool qualification is defined as, "The process necessary to obtain certification credit for a software tool within the context of a specific airborne system." It is the certification authority that decides on the outcome of the qualification process. Moreover, qualification, if claimed, is a requirement in getting a system certified.

Types of Software Development Tools

DO-178B differentiates between verification tools *that cannot introduce errors but may fail to detect them* and development tools *whose output is part of airborne software and thus can introduce errors*. There is a significant amount of effort involved to qualify a verification tool, and much more to qualify a development tool. However, numerous development tools have been used successfully in many certified projects without being qualified. To define a subject matter more narrowly, we need to take a closer look at the entire domain of software development tools.

The landscape of modern software development tools is very broad, as illustrated in Figure 1 (see page 20). Following the traditional model of the development process from requirements to implementation, we can identify the following:

- The *requirements* category that includes tools used early in the life cycle to identify and specify the software requirements.
- The *design* category that includes tools allowing developers to create architectural and detailed design of the software in a notation of their choice supported by the tool; often in this category, tools translate the model to source code.
- The *implementation* category that includes all support required to translate the computer code and transfer it to the target computer.

As illustrated in Figure 1, three other categories of tools can be identified: those related to *analysis*, *testing*, and *target*.

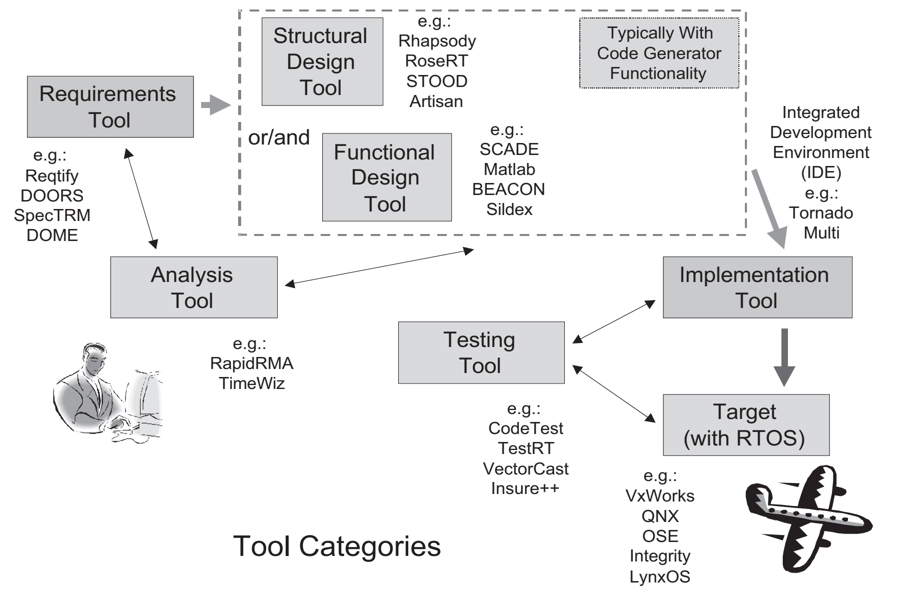

Requirements Tool
e.g.: Reqtify DOORS SpecTRM DOME
Structural Design Tool
e.g.: Rhapsody RoseRT STOOD Artisan
or/and
Functional Design Tool
e.g.: SCADE Matlab BEACON Sildex
Typically With Code Generator Functionality
Integrated Development Environment (IDE) e.g.: Tornado Multi
Analysis Tool
e.g.: RapidRMA TimeWiz
Implementation Tool
Testing Tool
e.g.: CodeTest TestRT VectorCast Insure++
Target (with RTOS)
e.g.: VxWorks QNX OSE Integrity LynxOS
Tool Categories

Figure 1: *Software Tool Categories*

However, for this article, we will focus primarily on the tools used in the design phase, the central component of the software development life cycle. They reflect two diverse viewpoints on real-time, safety-critical systems development, which result from different developers' backgrounds:

- Control engineers consider a system to be a dynamic model consisting of well-defined blocks of specific functionality (logic, arithmetic, dynamic). The functional paradigm of the model is the basis for system simulation and analysis of its behavior. Subsequently, the model can be translated *automatically* into an equivalent code, typically without any additional developer's involvement.
- Software engineers, on the other hand, are familiar with the concepts of operating systems, programming languages, software development methodologies, and notations. The graphic notations (classes, packages, states, transitions, events) allow developers to represent the structure and behavior of the target system software as a set of components that can be translated into programming constructs (data structures, objects, functions, etc.) using the automatic code generation functionality of the tool.

Consequently, the software design tools, which assist developers in translating the software requirements into source code, can be categorized into two groups: (a) a *function-based*, *block-oriented* approach applied by control and system engineers, and (b) a *structure-based*, *object-oriented* approach applied by computer scientists and software engineers.

The Qualification Process

A typical use for a design tool (software *producer*) is to transform an input artifact into output, thus creating another software artifact. The current process mandates verification after each transformation. If this transformation has an impact on the final airborne product, the producer needs to be qualified, but only if the transformation output would not be verified and the transformation leads to elimination, reduction, or automation of any of the DO-178B processes. The conditions under which a development tool requires qualification are presented in Figure 2 [4].

Software development tool qualification is attempted only as an integral component of a specific application program requiring the FAA's certification. The software tools to be used are referenced within the Plan for Software Aspects of Certification (PSAC) and the Software Accomplishment Summary documents of the original certification project. If development tool qualification is required, the applicant should present for review the Tool Operational Requirements (TOR) – a document describing tool functionality, environment, installation, operation manual, development process, and expected responses (also in abnormal conditions).

Two documents must be submitted and approved: a Tool Qualification Plan, and a Tool Accomplishment Summary as described in [4]. To make an argument for qualification, the applicant must demonstrate correctness, consistency, and completeness of the TOR and show that the tool complies with its TOR. This demonstration may involve a trial period during which a verification of the tool output is performed and tool-related problems are analyzed, recorded, and corrected.

Other data required for review include a Tool Configuration Management Index, Tool Development Data, Tool Verification Records, Tool Quality Assurance Records, Tool Configuration Management Records, etc. These requirements are also described in [4]. Tool qualification data are approved only in the context of the overall software development for the specific system where the intention to use the tool is stated in the PSAC. The tool itself does not receive a separate *qualification stamp of approval*. Therefore, using the tool on another system/project requires a separate qualification, although some qualification credits may be reused.

Surveys requesting which tools are used by industry were conducted at two national conferences: the 2002 FAA National Software Conference and the 2004 Embry Riddle Aeronautical University/FAA Software Tool Forum. In addition, two follow-up e-mail solicitations were sent to more than 500 professionals working on airborne systems. These surveys and solicitations resulted in a relatively small sample of responses that did not provide a base for statistically significant results. The comments included industry discouragement regarding the rigor of development tool qualification, and a justified perception of the extensive cost of qualification.

Potential solutions to assist in commercial off-the-shelf (COTS) development tool qualification included extensive vendor collaboration and using *alternate means* allowed in DO-178B. The limited feedback shows that there has been interest in qualifying software development tools classified in the function-based/block-oriented category, which cannot be said about structure-based/object-oriented tools.

A short list of qualified development tools includes code generators (Generation Automatique de Logiciel Avionique, Graphical Processing Utility, Virtual Application Prototyping System Code Generator, Safety Critical Avionic Development Environment Qualifiable Code Generator) and configuration-scheduling table generators (Universal Table Builder Tool, Configuration Table Generation Tool), most of them being in-house products. According to several informal exchanges with industry, many of the modern COTS software development suites actually have been used in the creation of software artifacts on certified projects without going through the qualification process.

Problems With Development Tool Qualification

It is clear that qualification of development tools is an option rarely exercised in the airborne software industry. In fact, one could argue that qualification of development tools is not a viable option. Current interpretation of applicable guidelines makes development tool qualification a proposition that is not practical from a *managerial viewpoint*, and not easy from a *technical viewpoint*.

Managerial Viewpoint

The first group of problems is of a regulatory and managerial nature. The major hurdle is the current state of regulations and guidelines. The secondary obstacle is the business model and lack of incentives, in particular the prohibitive cost of tool qualification. The existing tools, often used in certification projects, do not have appropriate data to support arguments about meeting the objectives of DO-178B. The applicant team's intent is to certify the product rather than expand effort and qualify the tool. The tool vendor does not see the business advantage of qualifying a tool while disclosing proprietary information to potential competitors.

Development tool qualification requires close collaboration between the tool vendor and the applicant. This is the reason why in-house tools are more likely to be qualified. Internal trade studies [5] have shown that the cost of development tool qualification is significantly higher than the cost of verification tool qualification. The use of qualified verification tools can result in fast savings on the first program where they are introduced. In contrast, the use of qualified development tools may require several programs to make up the cost.

The intellectual property rights may need to be waived by the vendor to achieve qualification. The tool cannot be qualified as standalone, but only within the scope of a particular certification project. The tools that could be considered for qualification are very simple: typically in-house created utilities where the applicant holds all intellectual property rights, maintains all tool development data, and can reuse the tool software artifacts on consecutive projects. The qualification is accomplished within the specific certification project and thus is not clearly visible from the outside as *development tool qualification*.

Technical Viewpoint

The second group of problems is related to technical aspects. According to the DO-178B interpretation, the development tool needs to be qualified to the same level of scrutiny as the appropriate application it is helping to develop. However, there is a significant difference between tool software and application software. Applications run on a target computer while tools operate on a general-purpose workstation, typically closely interacting with a COTS operating system and conventional programming environment. Considering this, several DO-178B objectives are not applicable to tool software and thus cannot be met. There is also no general agreement on what metrics would allow developers to carry an independent tool assessment [6].

One often-repeated statement regarding development tool qualification is the requirement that "only deterministic tools can be qualified." The DO-178B refers to determinism as "… tools which produce the same output for the same input data when operating in the same environment." The definition does not take into account how the output is generated. By this definition, one may interpret that it is not required to provide proof on the internal behavior of a tool. An example of this can be memory use for a tool running on the host workstation in a multitasking, multiuser, networked environment. The problem is to define what the object code for a tool is. Does it include the operating system (OS) of the host workstation? A tool clearly needs to make explicit calls to the OS routines, and any verification of these would require full visibility of the host's OS and related high assurance of its operation.

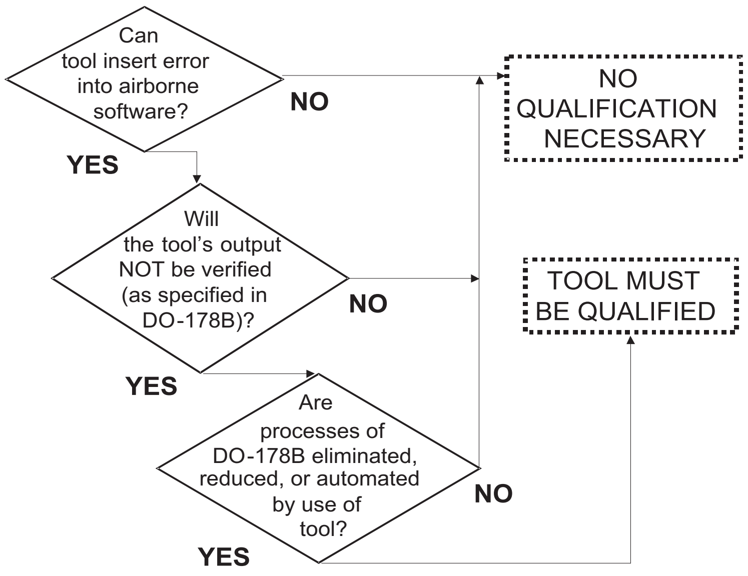
The main function of a software development tool is to transform, i.e., translate an input artifact into output. This is why the qualification, if applicable, should be focused on this translation component of the tool functionality. However, modern, complex software development tools provide a variety of other functions that are not directly related to the translation process. The translation component is hidden deep inside the tool, which causes problems with tool qualification. Typically, there is no access to a COTS tool's life-cycle data, which describe the tool's requirements, design, and code. Unless the tool has been developed in-house, the qualification efforts may be doomed.

Potential Solutions

The qualification of a stand-alone development tool is not feasible in the strict sense of existing guidelines. Such concepts as component-based software, software reuse, and service history should be explored [7] to identify the feasibility of such qualification. The issues of tool version control and the precise definition of operational environment, constraints, and limitations are the basis for starting discussion about solutions to tool qualification. The availability of extensive tool software development data, often scarce for COTS products, may be a challenge to ever accomplish COTS tool qualification [8].

It could be conceivable to create an independent lab dedicated to tool qualification and encourage commercial vendors to submit their product for assessment. A similar approach is known from other areas of verification and validation [9, 10]. Another idea would be to require certified product applicants to disclose information regarding the development tool use and qualification effort by creating an FAA-sponsored database for DO-178B certified products. This could face serious objections from industry due to an apprehensiveness to disclose any information, which may result in the loss of commercial advantage. It would be possible to research a potential for development tool

Figure 2: *Conditions When a Software Development Tool Requires Qualification*


Can tool insert error into airborne software?
NO
YES
NO QUALIFICATION NECESSARY
Will the tool's output NOT be verified (as specified in DO-178B)?
NO
YES
TOOL MUST BE QUALIFIED
Are processes of DO-178B eliminated, reduced, or automated by use of tool?
NO
YES

qualification using an approach different than the one outlined in Section 12.2 of DO-178B. Service history and formal methods could both be potential options.

It appears that the industry has a pressing need to come up with methods to audit a tool that is independent of the specific program and applications using it. This would require updating the guidelines to consider a model-driven development paradigm, redefine the qualification process, and allow flexibility regarding qualification to be less dependent on the application program using the tool. A more streamlined method to qualify development tools and to keep them current as technology advances would be useful. Better guidance on how to apply service history and how to address what has to be done for incremental tool changes would also be needed. These and other issues have been discussed at the recent Tools Forum [11]. The RTCA convened another special committee (SC-205) with a charge to recommend modifications to the existing DO-178B. The qualification of software tools is being discussed and some changes may be forthcoming.◆

Acknowledgement

The presented work was supported in part by the Aviation Airworthiness Center of Excellence under contract DTFA-0301 C00048 sponsored by the FAA. Findings contained herein are not necessarily those of the FAA. Additional support was received from the Florida Space Grant Consortium under Grant No. UCF01-E000029751.

References

1. Radio Technical Commission for Aeronautics, Inc. "RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification." Advisory Circular. Washington, D.C.: RTCA, 1 Dec. 1992 <www.rtca.org/downloads/ListOfAvailableDocsAPR%202005.htm#_Toc101071800>.
2. Federal Aviation Administration. "RTCA Inc., Document RTCA/DO-178B." Advisory Circular No. 20-115B. Washington, D.C.: U.S. Department of Transportation, Nov. 1993 <www.airweb.faa.gov/Regulatory_and_Guidance_Library/rgAdvisoryCircular.nsf/0/DCDB1D2031B19791862569AE007833E7?OpenDocument>.
3. Radio Technical Commission for Aeronautics, Inc. "RTCA DO-248B, Final Report for Clarification of DO-178B 'Software Considerations in Airborne Systems and Equipment Certification'." Advisory Circular. Washington, D.C.: RTCA, 10 Dec. 2001 <www.rtca.org/downloads/ListOfAvailableDocsAPR%202005.htm#_Toc101071717>.
4. Federal Aviation Administration. "Software Approval Guidelines." FAA Order 8110.49. Washington, D.C.: FAA, 2003 (Chapter 9 replaces FAA Notice N8110.91 of 2001) <www.airweb.faa.gov/Regulatory_and_Guidance_Library/rgOrders.nsf/0/640711B7B75DD3D486256D3C006F034F?OpenDocument&Highlight=8110.49>.
5. Potter, Bill. "Use of the MathWorks Tool Suite to Develop DO-178B Certified Code." Slide No. 13. Honeywell, May 2004 <http://faculty.erau.edu/korn/ToolForum/potter.htm>.
6. Kornecki A., and J. Zalewski. Criteria for Software Tools Evaluation in the Development of Safety-Critical Real-Time Systems. Proc. of PSAM-7/ European Safety and Reliability Conference, Berlin, Germany, 14-18 June 2004. London: Springer-Verlag, 2004 <http://faculty.erau.edu/korn/papers/ESREL04KorneckiZalewski.pdf>.
7. Lougee, H. "DO-178B Certified Software: A Formal Reuse Analysis Approach." CROSSTALK Jan. 2005 <www.stsc.hill.af.mil/crosstalk/2005/01/0501lougee.html>.
8. Zalewski, J., W. Ehrenberger, F. Saglietti, J. Gorski, and A. Kornecki. "Safety of Computer Control Systems: Challenges and Results in Software Development." Annual Reviews in Control 27.1 (2003): 23-37.
9. Brosgol, B.M. "ADA in the 21st Century." CROSSTALK Mar. 2001 <www.stsc.hill.af.mil/crosstalk/2001/03/brosgol.html>.
10. Adams, M., et al. "Conformance Testing of VMEbus and Multibus II Products." Advanced Multi-Microprocessor Bus Architectures. Ed. J. Zalewski. Los Alamitos, CA.: IEEE Computer Society Press, 1995: 392-399.
11. Embry Riddle Aeronautical University/FAA Software Tools Forum, Embry Riddle Aeronautical University, Daytona Beach, FL., May 18-19, 2004 <www.erau.edu/db/campus/softwaretoolsforum.html>.

About the Authors

Andrew J. Kornecki, Ph.D., is a professor at the Department of Computer and Software Engineering, Embry Riddle Aeronautical University. He has more than 20 years of research and teaching experience in areas of real-time computer systems. Kornecki contributed to research on intelligent simulation training systems, safety-critical software systems, and served as a visiting researcher with the Federal Aviation Administration (FAA). He has been conducting industrial training on real-time, safety-critical software in medical and aviation industries and for the FAA Certification Services. Recently, he has been engaged in work on certification issues and assessment of development tools for real-time, safety-critical systems.

Dept. of Computer and Software Engineering
Embry Riddle Aeronautical University
600 Clyde Morris BLVD
Daytona Beach, FL 32114
Phone: (386) 226-6888
Fax: (386) 226-6678
E-mail: kornecka@erau.edu

Janusz Zalewski, Ph.D., is a professor of computer science at Florida Gulf Coast University. Prior to this, he worked for various nuclear research institutions, including the Data Acquisition Group of Superconducting Super Collider and Computer Safety and Reliability Center at Lawrence Livermore National Laboratory. He also worked on projects and consulted for a number of private companies, including Lockheed Martin, Harris, and Boeing. Zalewski served as a chairman of the International Federation for Information Processing Working Group 5.4 on Industrial Software Quality, and of an International Federation of Automatic Control Technical Committee on Safety of Computer Control Systems. His major research interests include safety-related, real-time computer systems.

Dept. of Computer Science
Florida Gulf Coast University
10501 FGCU BLVD
Fort Myers, FL 33965
Phone: (239) 590-7317
Fax: (239) 590-7330
E-mail: zalewski@fgcu.edu